웹시스템설계 실습문서 Lab 10

최지헌 unidev@ajou.ac.kr
이재현 dlwogus8888@ajou.ac.kr

2024-11-26

목차

1 학습 목표

Mongoose Schema 설계 및 validation을 구현할 수 있으며, 데이터 모델간 관계 설정과 populate 활용 방법을 습득한다. 이를 통해 실제 서비스와 유사한 데이터 모델링을 경험한다.

2 MongoDB

- MongoDB란 Document 기반의 No-SQL Database이다.
- Document는 MongoDB의 데이터 저장 단위로, 한 개 이상의 Key-Value 쌍으로 이루어진 객체를 BSON 형태로 저장한다.
- SQL expression이 존재하지 않으며, 모든 데이터 조작은 mongoshell 또는 ODM(Object Data Mapping) 라이브러리에서 메소드 호출 방식으로 동작한다. Schema-free한 데이터 저장 방식 덕분에 같은 Collection에 속한 Document라도 다른 유형의 데이터 필드를 가질 수 있다 모든 Document는 기본적으로 **_id** 라는 고유한 값을 가진 필드가 존재한다.

Document는 MongoDB의 데이터 저장 단위로, 한 개 이상의 Key-Value 쌍으로 이루어진 객체를 BSON 형태로 저장한다. **Collection**은 데이터를 용도에 따라 분류하기 위한 Document의 모음이며, 모든 Document는 특정 Collection에 속한다. (RDBMS의 Table과 유사함) **Database**는 여러 Collection들을 담고 있는 물리적 저장소이다. MongoDB 서버는 여러 개의 Database를 가질 수 있으며, "System" 은 서버 운영에 필요한 메타 정보를 저장하기 위해 예약되어 있으므로 사용 불가하다.

2.0.1 Mongoose

Mongoose는 Node.js 기반의 MongoDB ODM 라이브러리이다. MongoDB Driver가 기본적으로 제공하는 CRUD 기능을 포함하여 다양한 추가 기능 제공한다.

- Schema (https://mongoosejs.com/docs/guide.html)
  - Schema 기반의 Collection 데이터 모델링이 가능하고, 데이터를 DB에 저장하기 전에 검사 가능
- Population (https://mongoosejs.com/docs/populate.html)
  - Document B를 참조(reference)하고 있는 Document A를 쿼리할 때, 참조 위치에 실제 Document B의 데이터가 치환되어 값이 반환되는 기능
- Callback-base / Promise-base 방식을 모두 지원하는 유연한 API 제공

2.0.2 Mongoose를 사용해 MongoDB 연결

Mongoose는 NPM을 통해 프로젝트에 추가할 수 있다.

```
$ npm install mongoose
```

MongoDB 서버와 통신하기 위한 connection 객체를 생성하여 연결한다.

```
async function connect(){
await mongoose.connect('mongodb://127.0.0.1:27017/test_db',
                    { useUnifiedTopology: true, useNewUrlParser: true })
}

connect();
```

서버 주소는 mongodb://ADDRESS:PORT/DATABASE의 형태로 기술한다. mongoDB는 따로 설정을 수정하지 않았다면 27017번 포트로 접근할 수 있다. 같은 컴퓨터 내에 설치되어 있는 mongoDB에 접근한다면 주소로 127.0.0.1이나 localhost를 사용한다.

2.0.3 Schema 정의

스키마(schema)란 collection에 저장할 document의 데이터 구조를 모델링하는 기능이다.

```
const userSchema = new mongoose.Schema({
    name: {
        first: String,
        last: String
    },
    createdAt: {
        type: Date,
        default: Date.now
    }
})
```

필드 이름과 속성으로 구성되며 속성으로 타입만 정의한다면 하위 객체를 넣을 필요 없이 바로 타입값을 넣을 수 있다. 사용 가능한 속성들로는 다음과 같은 것들이 있다.

- type: 이 필드의 타입을 지정한다. 값으로 자바스크립트의 타입 객체들이 온다.
- unique: 해당 collection 내에서 document 간 중복되지 않는 필드임을 명시한다. 이 필드가 중복되는 상황을 만드는 document 추가나 수정은 에러가 발생하게 된다. 값으로 boolean값이 온다.
- required: 필수값임을 명시한다. 이 값 없이 document를 생성한다면 에러가 발생한다. 값으로 boolean 값이 온다.
- default: document를 생성하여 저장할 때 이 필드의 값이 존재하지 않는다면 기본적으로 주어질 기본값을 정의한다. 값으로 아무것이나 올 수 있다.

2.0.4 Model 생성

모델(model)이란 앞서 정의한 스키마를 기반으로 collection에 조작을 가하기 위한 인터페이스 객체이다.

```
const userModel = mongoose.model('User', userSchema)
```

여기서 model 메서드의 첫 번째 파라미터로 주어진 "User"가 collection의 이름이 된다. 단수로 지정할 경우 자동으로 s가 붙어 복수 형태로 collection 이름이 지정된다.

2.0.5 CRUD Example (Create)

```
const data = await userModel.create({ name: { first: "aa", last: "bb" } })
const res = await data.save();
console.log(res);
```

2.0.6 CRUD Example (Read)

```
const result = await userModel.find({name: { first: "aa", last:"bb" }});
```

2.0.7 CRUD Example (Update)

조건에 맞는 모든 document의 필드값을 갱신할 수 있다. updateMany(조건, 갱신값)의 형태로 기술된다.

```
const result = await userModel.updateMany({name: { first: "aa", last:"bb" }}, {name: {last:
    "cc"}});
```

응답 객체로 필드값으로 n과 nModified, ok가 담긴 객체가 반환된다. n은 선택된 document의 총 개수, nModified는 실제로 수정된 document의 개수이며 ok는 성공 여부이다. 조건에 일치하는 것 중 첫 번째 것만 업데이트하기 위한 메서드로 updateOne이 있다. 갱신값에는 기술된 필드만 갱신되고 기술되지 않은 필드는 기존값 그대로 남는다. 완전히 document를 덮어씌우려면 replaceOne를 사용한다.

2.0.8 CRUD Example (Delete)

조건에 맞는 모든 document를 제거한다. deleteMany(조건)의 형태로 기술된다.

```
const result = await userModel.deleteMany({name: { first: "aa", last:"bb" }})
```

조건에 일치하는 것 중 첫 번째 것만 제거하는 api로 deleteOne이 존재한다.

3 실습 과제

3.1 과제 개요

유저 기능을 제외한 게시글 및 댓글 작성을 위한 백엔드 시스템을 구현한다. 게시글과 댓글을 작성할 수 있다.

3.2 Model Specification

3.2.1 데이터 모델 구조

Post Schema 게시물의 기본 정보를 저장하는 주요 모델이다.

필드명	타입	필수여부	제약조건
title	String	Required	minlength: 2, maxlength: 100
content	String	Required	minlength: 10
author	String	Required	minlength: 2, maxlength: 20
category	String	Required	enum: ['tech', 'life', 'food'...]
tags	[String]	Required	validate: array.length 1-5
views	Number	Auto	default: 0
createdAt	Date	Auto	default: Date.now
updatedAt	Date	Auto	default: Date.now

Comment Schema 게시물에 종속된 댓글 정보를 저장한다.

필드명	타입	필수여부	제약조건
content	String	Required	minlength: 2, maxlength: 500
author	String	Required	minlength: 2, maxlength: 20
post	ObjectId	Required	ref: 'Post'
parent	ObjectId	Optional	ref: 'Comment'
depth	Number	Required	max: 1 (대댓글 제한)
createdAt	Date	Auto	default: Date.now
updatedAt	Date	Auto	default: Date.now

3.3 기능 요구사항

3.3.1 게시물 관리 기능

게시물 작성 (POST /api/posts)

- 제목, 내용 길이 검증
- 작성자명 길이 검증
- 카테고리 유효성 검증
- 태그 개수 검증 (1-5개)

게시물 조회 (GET /api/posts/:id)

- 조회수 자동 증가
- 연관 댓글 목록 포함
- 작성일, 수정일 포맷팅

게시물 수정 (PUT /api/posts/:id)

- updatedAt 자동 갱신
- 부분 수정 지원 (title, content, category, tags)

3.3.2 댓글 관리 기능

댓글 작성 (POST /api/posts/:id/comments)

- 내용 길이 검증 (2-500자)
- 대댓글의 경우 부모 댓글 ID 검증
- depth 검증 (최대 1단계)

댓글 수정/삭제

- 작성자명 일치 여부 확인
- 삭제 시 하위 댓글도 함께 삭제
- soft delete 구현 (isDeleted 플래그 사용)

3.3.3 검색 및 필터링 기능

게시물 검색 (GET /api/posts)

- 태그 기반 필터링
- 정렬 옵션 지원 (최신순, 조회수순)

3.4 제출 요구사항

- 제출 기한: 11월 26일 자정 / 지각 제출: 11월 28일 자정
- node_modules 디렉토리를 제외하고 프로젝트 전체를 압축
- 압축 파일명: Lab10_학번.zip